Date of acceptance Grade

Instructor

Bayesian Methods for Prediction of Atomic Migration Barriers for Quantum-Mechanics-Based Material Design

Carlos Santana Vega

Helsinki June 21, 2018

Master Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Carlos Santana Vega

Työn nimi — Arbetets titel — Title

Bayesian Methods for Prediction of Atomic Migration Barriers for Quantum-Mechanics-Based Material Design

Oppiaine — Läroämne — Subject
Data Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master Thesis	June 21, 2018	33 pages + 0 appendices

Tiivistelmä — Referat — Abstract

The scope of this project is to provide a set of Bayesian methods to be applied to the task of potential energy barriers prediction. Energy barriers define a physical property of atoms that can be used to characterise their molecular dynamics, with applications in quantum-mechanics simulations for the design of new materials. The goal is to replace the currently used artificial neural network (ANN) with a method that apart of providing accurate predictions, can also assess the predictive certainty of the model. We propose several Bayesian methods and evaluate them on this task, demonstrating that sparse Gaussian process (SGP) are capable of providing predictions, and their confidence intervals, with a level of accuracy equivalent to the current ANN, in a bounded computational complexity time.

Avainsanat — Nyckelord — Keywords
bayesian methods, quantum mechanics, material design, sparse Gaussian process

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

"I can live with doubt and uncertainty and not knowing. I think it is much more interesting to live not knowing than to have answers that might be wrong. [...]"

- Richard Feynman.

In recent years, we have experienced an explosion on the application of advanced deep learning techniques in many different contexts (e.g. autonomous driving, robotics, health-care, business, etc.). Keys behind this new renaissance in the machine learning field are the development of novel and more advanced algorithms; the fast evolution in the market of general-purpose computing chips; and the exponential growth of stored structured data, as a consequence of the massive digitalisation of our societies.

In this new wave of AI, neural networks have become ubiquitous, representing a new computational framework capable of deducing the underlying complex relationship between variables, just by learning from the given observations. However, associated with their powerful predictive capabilities, neural networks are also known by their lack of interpretability.

Internally, a neural network constructs a high-dimensional representation of the given data, defined by the millions of parameters adjusted during the training process. The complexity of these systems makes hard to obtain insights about the chain of reasoning behind predictions and also to externalise the degree of uncertainty of those predictions. Because of this characteristic, neural networks are typically considered as black-boxes models [ST17].

In some scenarios, where reliable and certain measurements are required, this opaqueness in the predictions is undesirable, and alternatives to neural networks are required. In this work, we will focus on one of these scenarios: potential energy barriers predictions for large-scale simulation of atomic migrations.

1.1 Project scope

This Master Thesis is a subproject of the Machine Learning for Quantum Mechanics-Based Material Design (MachQu) project. A collaborative project between the Helsinki Institute of Physics and the Department of Computer Science of the University of Helsinki.

The goal of this collaboration is to develop a system that can reliably predict the potential energy barriers for a given atomic configuration. Energy barriers can be used to predict migrations of atoms and perform large-scale simulations at the atomic-level. These simulations can be used to characterise different physical processes in new materials, in high energy conditions, such as the experienced in particle accelerators (e.g. LHC, CLIC) or fusion and fission reactors (e.g. ITER).

The current state of the MachQu project already implements a neural network that can predict with high accuracy the barrier values, developed by the Helsinki Institute of Physics [LJV+18a]. However, as it was mentioned, by using neural networks is not possible to obtain any measurement that explains what the uncertainty of the predictions is. Because of this, it is not possible to assure if the neural network has learned a correct representation of the barrier values for the given domain, or it is just random guessing. In the context of this project, introducing unreliable measurements to the atomic simulation would end in characterising unreal processes of the material behaviour. We expect that by identifying the uncertain predictions and computing them by more accurate methods, we can improve the performance of the developed simulator.

The scope of this work is to provide an alternative model that can approximate the predictive capabilities of the neural network, while at the same time, can also quantify the uncertainty of the predictions. From the wide range of models that can be applied, in this work, we will only focus on the Bayesian model's spectrum.

The provided model will be part of an active learning system that will perform online predictions of the barrier values at high speed (*fast cycle*) (see Fig. 1). To train this model, it is required to reliably calculate the potential barriers for a subset of all the possible configurations of the atoms. These calculations are obtained by applying the nudged elastic band (NEB) method [JMJ98], which can provide an accurate computation of barriers, but at a high cost in computational time (*slow cycle*).

The system will provide the predicted potential energy barriers to the simulator module, implemented by the kinetic Monte Carlo (KMC) method [AFHW91], and

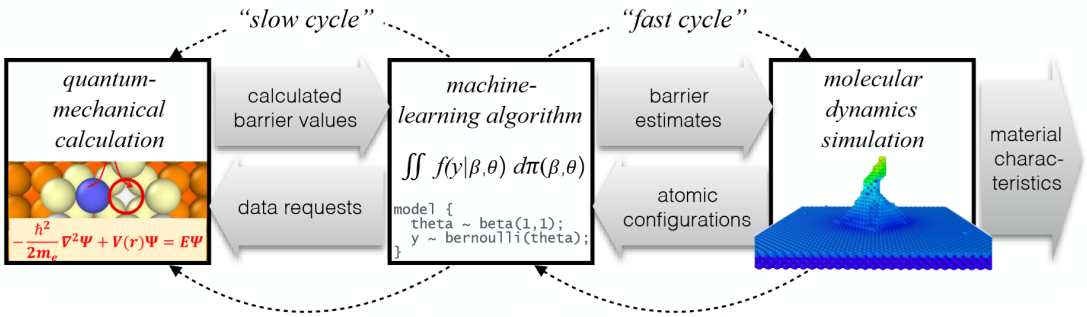

Figure 1: **Diagram of the proposed system. Figure courtesy of T. Roos and F. Djurabekova (unpublished project plan)**

it will provide a predictive model of long-term evolution of the atomic structure of complex materials. The KMC method will use the computed barriers to define a set of probabilities of atoms migrating to new positions and stochastically will decide in each iteration which atom will perform the transition. Once the atom moves to a new position, a new set of energy barriers should be recomputed for the local neighbourhood of atoms where the transition was performed. The simulator module will continue requesting iteratively to the designed probabilistic model for the barrier values of the new configurations, completing, therefore, the *fast cycle* loop.

As the online *fast cycle* continues predicting barriers, some atomic configurations could produce a prediction for which the model is very uncertain. The active learning module will use the uncertainty measurements of the model as a decision-making criterion to request a reliable calculation to the *quantum-mechanical calculation* layer, minimising the number of calls to the *slow cycle* only to the cases where predictions are unreliable.

Eventually, as the model gets trained by requesting the necessary observations, the system will be capable of mimicking the accurate calculations obtained by the NEB method, but in a fraction of it computational time.

The architecture proposed adds some requisites that should be considered when selecting a model. To avoid a bottleneck in the performance of the *fast cycle*, the model should perform predictions at high speed. Also, the model should provide online learning capabilities, so when the *quantum-mechanical calculation* layer calculates a new barrier, it should be possible to retrain the model with the new observation, in a reasonable time.

In this work, we will present some suitable alternatives to the current neural network model to be used in the presented system, and we will evaluate them both by their predictive capabilities and how well calibrated they are on their uncertainty measurements.

1.2 Structure of the thesis

The following sections of this document are organised as follows. In Sec. 2 we introduce all the theoretical aspects of this project, we define the problem to be solved and formally present the different models that will be compared.

Section 3 presents the results obtained from testing the different methods, providing more insights into those models that show the best performance.

Finally, in Sec. 4 we discuss the conclusions and how they align with the current and future state of the project.

2 Methods and Materials

In this section, we will present the Machine Learning problem addressed in this thesis. We also introduce all the theoretical framework explored during the research process.

2.1 Problem definition

For a given atom, we can use the potential barrier value to estimate the likelihood of the atom migrating to a new position. In this project, we perform a supervised learning process that predicts from the configuration of local neighbours atoms x_i,

the value of the potential barrier y_i. The value of the potential barrier, and therefore the likelihood of the migration, depends on the local environment of the migrating atom: both in the configuration of the local neighbours, and the nature of the atoms (e.g. copper, iron, gold, etc.). However, during this work, we are only limited to the scenario in which all the atoms in the configuration are of the same type.

The local neighbour configuration of a migrating atom corresponds to the set of atoms on the first and second level nearest-neighbours, laid out in a three-dimensional space. As can be seen in Fig. 2, we can index all the locations surrounding the migrating atom with a value that ranges from 0 to 25. Then, these locations can be encoded by a 26-dimensional binary array, so positions in the array correspond to the indexes atom location in the configuration, and the binary values stored in the array will specify if an atom occupies that position or not.

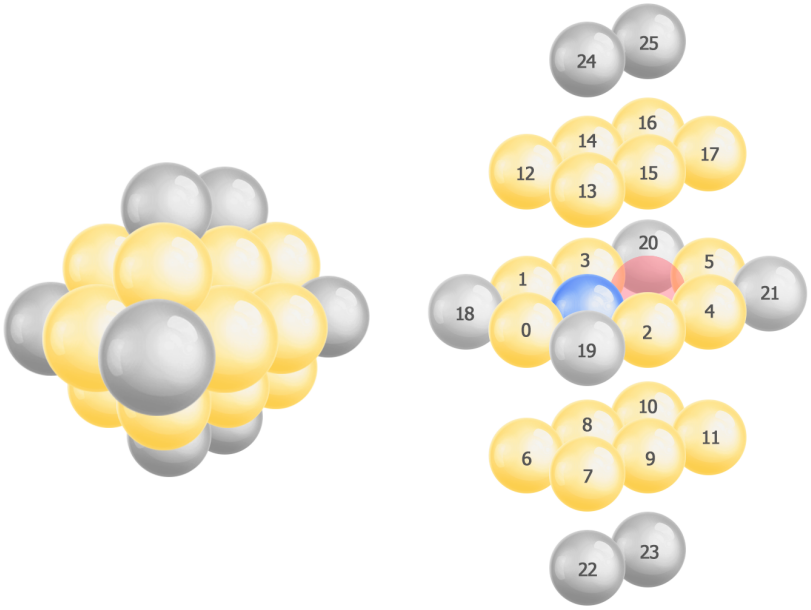

Figure 2: **Local neighbourhood of a migrating atom (blue) to a new location (red). Figure adapted from [LJV+18a]**

Another property that defines atoms configurations is its surface structure. This property, explains how the of the crystal structure of atoms are arranged depending on the orientation of the plane used to truncate their exposed surface. The given dataset is structured based on this property, so there are three different splits that

separate configurations depending on their surface structure. Following the Miller index notation, as explained in [Her16], these Copper surface structures are identified by the family type {100}, {110} and {111}.

The predicted values y correspond to energy barriers that the target atom should overcome to transition to a new position, in the given configuration. As energy magnitudes, these values are positive and range approximately between 0 and 2eV. Find in Fig. 3 a normalised histogram of the energy barrier values, grouped by surface structures. As can be seen, each surface follows a slightly different distribution, and all of them present a large peak of zero values.

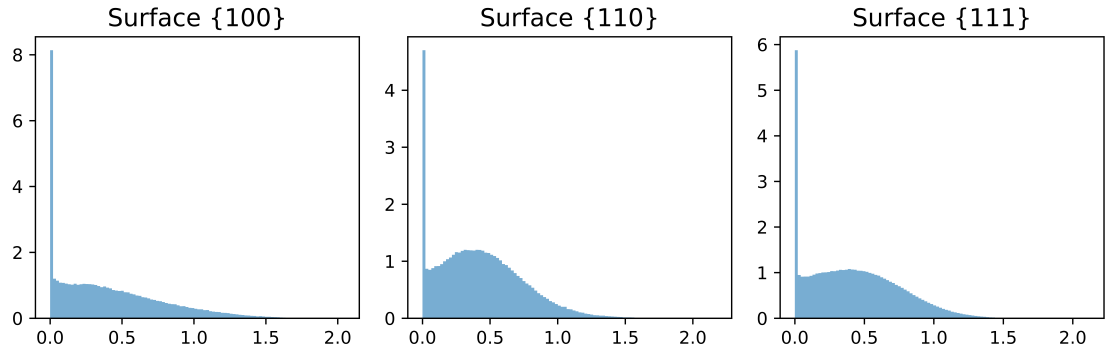

Figure 3: **Energy barrier histograms by surface structure.**

2.2 Bayesian Statistics

Contrary to classical frequentist statistics, Bayesian statistics links probabilities to uncertainties and parameters are not considered as fixed values, but random variables. The probability distributions of the parameters will reflect our degree of uncertainty on the real parameter values, and data will be used to update our prior beliefs, conditioning on observations, so we can compute a posterior distribution that will define our certainty about the parameters.

At the initial point of a Bayesian analysis, when no data has been observed, we can make some subjective assumptions on the possible values of the parameters w. These assumptions are encoded in the *prior distribution*:

$$P(w), \qquad w = (w_1, ..., w_p) \tag{1}$$

Depending on how strong are our a priori beliefs, we can decide whether to use an informative or an uninformative prior. Priors can be informative if they express strong assumptions about the possible values of the parameters. For instance, we can encode our assumption of a fair coin by selecting a distribution centred at 0.5. On the other hand, if we are not sure about whether the coin is fair or biased, then, we might prefer to choose a non-informative prior that assigns balanced likelihoods to each outcome and introduces minimal information about our subjective beliefs into the model. In both cases, the effect of the selected prior to the posterior distribution is dissipated when more observations are presented to the model. Also, as we will see in 2.3, selecting the correct prior distribution can also lead to some analytical benefits.

For a given data vector X, we can define what is the probability of the observations given the parameters value. If data points in D are independent and identically distributed ($i.i.d$), then the likelihood distribution can be factorized as:

$$P(X|w) = \prod_{i=1}^{n} P(x_i|w) \tag{2}$$

In combination, we use the prior distribution and the likelihood distribution to compute the distribution of the parameters conditioned to the observed data. This is called the posterior distribution and can be obtained through the application of Bayes Theorem:

$$P(w|X) = \frac{P(X|w)P(w)}{P(X)} \tag{3}$$

From the posterior distribution, it is possible to calculate point estimates such as MAP estimate, compute credible intervals of the parameters to measure their uncertainty or make predictions based on the whole distribution. Now, for a given new data vector x_{n+1}, the previous posterior distribution $P(w|X)$ will act as our prior function, and we will continue applying Bayes' Theorem to update our knowledge about the parameters of the model.

More interestingly, once we have encoded all the observations in the posterior, we can compute the distribution for predictions on new data, conditioned on the data we have already observed. This is known as the posterior predictive distribution:

$$P(x_{n+1}|X) = \int_w P(x_{n+1}|w)P(w|X)dw \tag{4}$$

In the presence of uncertainty in the parameters, the posterior predictive distribution averages over all the possible values of w, to define a distribution of the

predicted values for the new unobserved data. We can use this distribution to obtain the expected, predicted value, and its variance, to measure how confident we are about these predictions.

In this project, we will use the properties of the posterior predictive distribution to model the uncertainty of predictions. By measuring this uncertainty, we can detect whenever the model makes an unreliable prediction based on no evidence.

In Eq. 3, we see that computing the posterior distribution requires normalising the expression by the constant $P(X)$, the marginal likelihood The effect of this term is to scale the shape of the distribution, so its area adds up to 1. However, computing the marginal likelihood requires to marginalise out w from the joint distribution $P(X, w)$. This is expressed as the following integral:

$$P(X) = \int_w P(X, w)\, \mathrm{d}w \tag{5}$$

This marginalisation operation requires solving a high dimensional integral that for most of the models is intractable, and it usually is approximated by means of Markov Chain Monte Carlo (MCMC) methods. However, for the requirements of this project, deriving the posterior distribution using sampling methods is still computationally expensive. An alternative approach of sampling is to make use of conjugate priors, so the posterior distribution (and hence the posterior predictive distribution) can be derived analytically without heavy computations involved.

2.3 Conjugate priors

As discussed in the previous section, the prior distribution is the term in the Bayes' Theorem that encodes all our assumptions about the possible values of the parameters, before observing any evidence. Beyond its epistemological properties, we can also obtain computational benefits from selecting a right prior. A particular type of functions, denoted as conjugate priors, leads to analytical expressions of the posterior distribution that avoids the intractable derivations with integrals.

We say that a prior $P(w)$ is conjugated of a likelihood function $P(X|w)$, when we can combine them to compute a posterior distribution $P(w|X)$ that belongs to the same family of distributions as the prior.

For example, it is proven that for a Binomial likelihood function, the Beta distribution is a conjugate prior. Therefore, we can expect that the resulting posterior is

also a Beta distribution, defined by a new parametrisation that includes both the information encoded in the prior and the new data observations.

The utility of using conjugate priors is that, once we know what the functional form of the posterior distribution is, the training process only requires evaluating the expressions derived analytically for the new data observations.

Because the posterior and the prior distribution share the same functional form, it means that for the upcoming new data, we can use the previous posterior as our new prior distribution, with guarantees that the form of the posterior will always belong to the same family. This process of iteratively chaining computations as data comes in, is denoted as online learning and conjugate priors offer a suitable analytical solution to perform these updates during the process, that would be computationally intractable by other means.

For this reason, part of this project focuses the attention on the research of suitable conjugate Bayesian model that can be applied to our task.

2.4 Active Learning

Supervised Machine Learning methods look for the mapping function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^p$ is the input space and $\mathcal{Y} = \mathbb{R}$ is the output space. This function is approximated based on a given training set of samples $D = \{(x_1, y_1), ...(x_n, y_n)\} \in (X \times y)^n$, which defines pairs of input vectors $x_i \in X$ with its corresponding labelled outputs $y_i \in y$. These methods typically require large amounts of labelled data to perform both training and hyperparameter optimisation.

However, in many contexts, the access to this data is limited, either by the nature of the problem or because of the lack of resources (time and money). In this type of situations, we can take advantage of active learning strategies.

In an active learning set, we can reduce the amount of required labelled data used, by letting the algorithm to query to an oracle (human or a more advanced model) to label the data points that it considers necessary to improve its training. So instead of providing a full range of data points, now we can let the model to actively decide what observations needs to be labelled and given for training, so the uncertainty of the model can be reduced.

With this framework, we can reduce both the amount of time, as we let the model learn faster by using the most explicative data, and cost, since less data-labeling is

required. In the context of our project, we will apply active learning to minimise the number of queries to the slow process of performing the quantum-mechanical calculation.

To perform active learning, different strategies have been proposed [Set09]. To select the more informative query, an implementation based on *query-by-committee* set up a committee of models trained with the same dataset. To select the query, the models are asked to predict unlabeled data, and the data point with a higher degree of discrepancy is the one requested to the Oracle.

A different approach is *expected model change*, in which the model asks for those points that are expected to be more disruptive for the current state of the model. In a similar idea,*expected error reduction* methods target for those points that the model estimates are going to reduce the generalisation error the most.

However, the most used approach for active learning is *uncertainty sampling*, in which we ask the Oracle to label observations with higher predictive variance.

Uncertainty can be modelled in different ways, but given that in this project we follows the Bayesian framework, model uncertainty can quantified from the *predictive posterior distribution*. From a given posterior predictive distribution $P(y_{n+1}|x_{n+1}, y, X)$, a possible *active learning* strategy could be:

1. Compute $P(y_{n+1}|x_{n+1}, y, X)$ for all unlabeled $x_{n+1} \in X$.
2. Pick x_q, whose $P(y_q|x_q, y, X)$ has the largest predictive variance.
3. Query to the Oracle the label y_q for the observation x_q.
4. Update the posterior $P(w|y, X)$ where $y := (y, y_q)$ and $X := (X, x_q)$.
5. Repeat from (1) using the new posterior distribution.

By iterating this procedure, we will eventually reach a predictive posterior distribution $P(y_{n+1}|x_{n+1}, y, X)$, with a lower variance, at minimum cost in terms of used labelled data points.

2.5 Bayesian linear regression

Linear regression can be considered one of the most used types of models nowadays. The reason is that they are powerful enough to discover the linear relationships

between many variables, and also they are simple to use and to interpret.

A more powerful version of this inference method is the *Bayesian linear regression*, which besides capturing all the linear relations in our data, also has the potential of shaping the uncertainty of the parameters and predictions from the posterior and predictive distributions.

In the following sections, we will explore different versions of *Bayesian linear regression* that also introduce the use of a conjugate prior. As we mentioned in the previous section, by setting a conjugated prior in our model, we can derive an analytical expression to update the parameters of the model (i.e. training process) in constant time $O(1)$.

2.5.1 Normal-Gaussian

The classical version of linear regression seeks for the parameter vector $w \in \mathbb{R}^p$ that, once multiplied to the input vector x, can map the linear relationship between the independent and dependent variables. We can obtain the likelihood function of this model by assuming the existence of an $i.i.d$ error term ϵ that introduces random noise into the predictions. The noise introduced to the model is defined by the parameter σ^2, which determines what is the variance of the output observations. Then, the likelihood distribution of the regression is defined as follows:

$$y = X^\top w + \epsilon \tag{6}$$

$$\epsilon \sim N(0,\, \sigma^2) \tag{7}$$

$$P(y|X, w) = N(X^\top w,\, \sigma^2 I) \tag{8}$$

To translate this model scheme to the Bayesian framework, first, we have to define a prior distribution for the model parameters. Following the derivation from [RW05], the standard *Bayesian linear regression* assumes that parameters follow a Gaussian distribution with a zero mean and Σ_p covariance matrix.

$$w \sim \mathcal{N}(0,\, \Sigma_p) \tag{9}$$

Then, we combine the prior and the likelihood function to derive the posterior distribution by applying Bayes' rule (Eq. 3). We can transform this equation so the resulting expression is the following:

$$P(w|X, y) \propto \exp(-\frac{1}{2}(w - \bar{w})^\top(\frac{1}{\sigma^2}XX^\top + \Sigma_p^{-1})(w - \bar{w})). \tag{10}$$

$$P(w|X,y) = N(\bar{w} = \frac{1}{\sigma^2}A^{-1}Xy, A^{-1}), \tag{11}$$

where $A = \sigma^{-2}XX^\top + \Sigma_p^{-1}$. From the obtained expression, we can identify that the form of the posterior distribution is again a Gaussian distribution, the same form as the prior. As we covered in 2.3, this is because the normal distribution is a conjugate prior for the linear regression model, and therefore, prior and posterior are expected to belong to the same family distribution.

Furthermore, given that the prior is conjugated, then, we can also expect the predictive posterior distribution to be also from the same family of distributions. The predictive posterior is obtained by averaging all the possible predictions of the linear regressions that can be generated from the posterior distribution. By solving Eq. 4, we can obtain an expression from which a normal functional form can be identified:

$$P(x_{n+1}|X) = N(\frac{1}{\sigma^2}x_{n+1}^\top A^{-1}Xy, x_{n+1}^\top A^{-1}x_{n+1}), \tag{12}$$

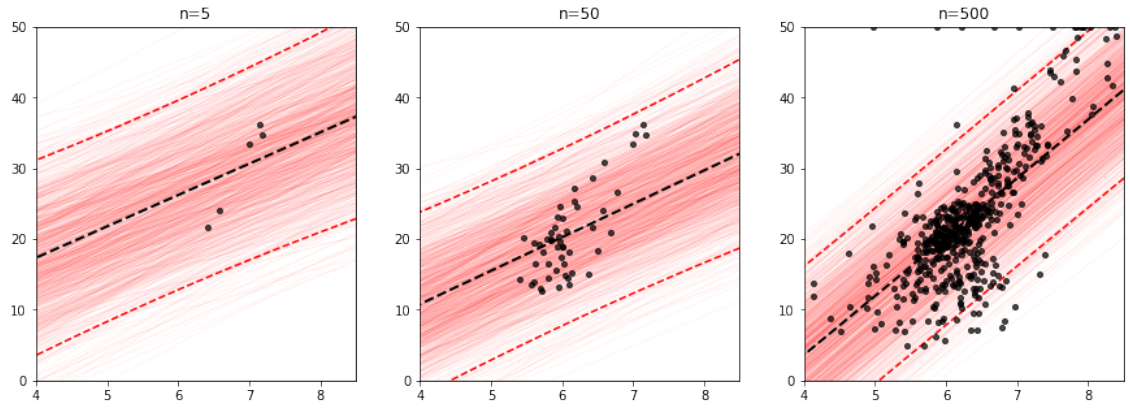

Figure 4: **Fixed-variance Bayesian Linear Regression with normal prior.**

We illustrate in Fig. 4 how the normal-Gaussian model applies to the Boston Housing dataset. We visualise (red dashed line) the 95% confidence level of the predictive posterior distribution for $n = 5, 50, 500$ samples. We can see how as the number of data points increases, the distribution aligns with the regression line and get a little narrower. However, the effect of the fixed variance $\sigma^2 = 80$, makes not possible to correctly estimate the predictive variance of the samples in the first two panels.

2.5.2 Normal-inverse-gamma

In the previous model, we assumed that the variance of the model is known and that the only parameter that we need to infer is the vector of weights w. However, it is possible to use the Bayesian framework to build a conjugate model that also captures the uncertainty of the parameter σ_2.

As we need to define a conjugate prior for the two unknown parameters, a valid option is to use the *normal-inverse-gamma (NIG) prior*.

This prior assumes that the vector of weights w are distributed as a normal, with mean μ_w and variance $\sigma_2\Lambda_0^{-1}$ where Λ_0 is a positive definite precision matrix of size $p \times p$. The variance σ_2 follows an *inverse-gamma* distribution defined by two positives hyperparameters a, b. The combination of this distribution it is known as the *normal-inverse-gamma* distribution, and can be denoted as $NIG(\mu, \Lambda, a, b)$:

$$P(w, \sigma^2) = P(w|\sigma^2)P(\sigma^2) = N(\mu_0, \sigma^2\Lambda_0^{-1})IG(a_0, b_0) = NIG(\mu_0, \Lambda_0, a_0, b_0). \quad (13)$$

Then, the posterior distribution is obtained by applying Bayes' rule (eq. 3):

$$\begin{aligned} P(w, \sigma^2|y, X) &\propto P(y|X, w, \sigma^2)P(w|\sigma^2)P(\sigma^2) \\ &\propto (\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}(y - Xw)^{\mathrm{T}}(y - Xw)\right) \\ &\quad (\sigma^2)^{-\frac{m}{2}} \exp\left(-\frac{1}{2\sigma^2}(w - \mu_0)^{\mathrm{T}}\Lambda_0(w - \mu_0)\right) \\ &\quad (\sigma^2)^{-(a_0+1)} \exp\left(-\frac{b_0}{\sigma^2}\right). \end{aligned} \quad (14)$$

Now we can proceed to obtain the update rules of the parameters. Based on the steps from Fahrmeir et al. (2007), as cited in [WA10], the obtained posterior can be transformed to an expression from which a new parametrization for the *NIG* distribution can be recognized:

$$P(w, \sigma^2|y, X) \propto NIG(\mu_n, \Lambda_n, a_n, b_n), \quad (15)$$

where the parameters are computed from the following update rules:

$$
\begin{aligned}
\mu_n &= (X^\top X + \Lambda_0)^{-1}(\Lambda_0\mu_0 + X^\top y), \\
\Lambda_n &= (X^\top X + \Lambda_0), \\
a_n &= a_0 + \frac{n}{2}, \\
b_n &= b_0 + \frac{1}{2}(y^\top y + \mu_0^\top \Lambda_0 \mu_0 - \mu_n^\top \Lambda_n \mu_n)
\end{aligned}
\tag{16}
$$

These expressions are the rules that we should compute to update the posterior distribution sequentially when training the model for new observations x_{n+1}.

To obtain the uncertainty of these new observations, we have to compute the predictive posterior distribution. Based on the steps from [WA10] we can see that in the NIG model, this distribution follows a *multivariate t* density with:

$$
\begin{aligned}
P(x_{n+1}|x) &= \int P(x_{n+1}|w,\sigma^2)p(w,\sigma^2|x)dwd\sigma^2 \\
&= MVSt_{2a_n}(x_{n+1}\mu_n, \frac{b_n}{a_n}(I + x_{n+1}\Lambda_n^{-1}{x_{n+1}}^\top))
\end{aligned}
\tag{17}
$$

$$
P(x_{n+1}|X) = N(\frac{1}{\sigma_n^2}x_{n+1}^\top A^{-1}Xy, x_{n+1}^\top A^{-1}x_{n+1}), \tag{18}
$$

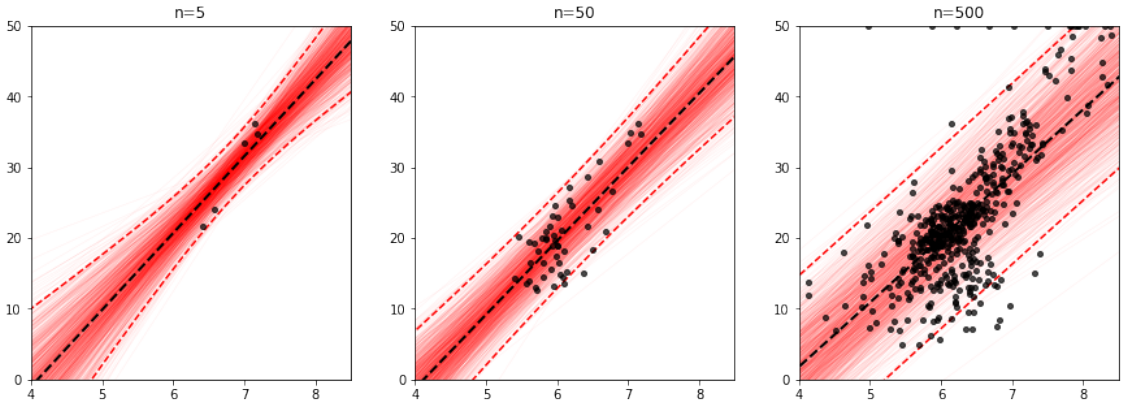

Figure 5: **Example of *NIG* predictive posterior distribution.**

As we previously did, we visualise in Fig. 5 the predictive posterior distribution for the *NIG* model. Now we can see how the model can estimate a more realistic predictive variance, where the noise from the data is captured.

2.5.3 Normal-inverse-Wishart

We can obtain a different parametrization for the *NIG* distribution if we consider its general formulation: the normal-inverse-Wishart *(NIW)* distribution. The use of *NIW* conjugate prior allows us to perform *multivariate Bayesian linear regression*, where instead of having just one dependent variable y_n, we can predict the values for a set of d dependent variables $y_{n,d}$. When $d = 1$, the problem is reduced to just *simple Bayesian linear regression* and the results obtained by applying *NIW* prior are equivalent to what is obtained when using *NIG*.

In the *NIW* model, the parameters assumed to be unknown are the regression mean μ and the covariance matrix Σ. Similarly as we did in the previous section, we can define the *prior distribution* of the model as the factorization of each marginal prior:

$$P(W, \Sigma) = P(W|\Sigma)P(\Sigma) = N(B_0, \Sigma \otimes \Lambda_0^{-1})IW(V_0, v_0) = NIW(B_0, \Lambda_0, V_0, v_0), \tag{19}$$

where $\otimes$ operation denotes the Kronecker product between two matrices.

The likelihood function $P(Y|X, W)$ for the *multivariate Bayesian linear regression* is defined as:

$$P(Y|X, W) = N(X^\top W, \Sigma), \tag{20}$$

where Y is a $n \times d$ matrix whose columns correspond to predictions on each of the d equations; W is a $p \times d$ dimensional weight matrix; and X is a $n \times p$ dimensional input matrix.

We can obtain the posterior distribution for the *NIW* model by applying Bayes rule (Eq. 3) to combine the two previous expressions. Because our prior is conjugated, we can follow the steps in [RAM06] to obtain an expression that can be translated to a new parametrization of the NIW distribution:

$$P(W, \Sigma|Y, X) \propto NIW(B_n, \Lambda_n, V_n, v_n). \tag{21}$$

where the parameters are obtained from the following update rules:

$$\begin{aligned} V_n &= V_0 + (Y - XB_n)^\top (Y - XB_n) + (B_n - B_0)^\top \Lambda_0 (B_n - B_0), \\ v_n &= v_0 + n, \\ B_n &= (X^\top X + \Lambda_0)^{-1}(X^\top Y + \Lambda_0 B_0), \\ \Lambda_n &= X^\top X + \Lambda_0 \end{aligned} \tag{22}$$

As we previously deduced, we can use these expressions to update the posterior distribution when new observations are introduced to the model. Visualizing in Fig. 6 the predictive posterior for the *NIW* model, we can confirm that, as expected, it behaves identically to the *NIG* model.

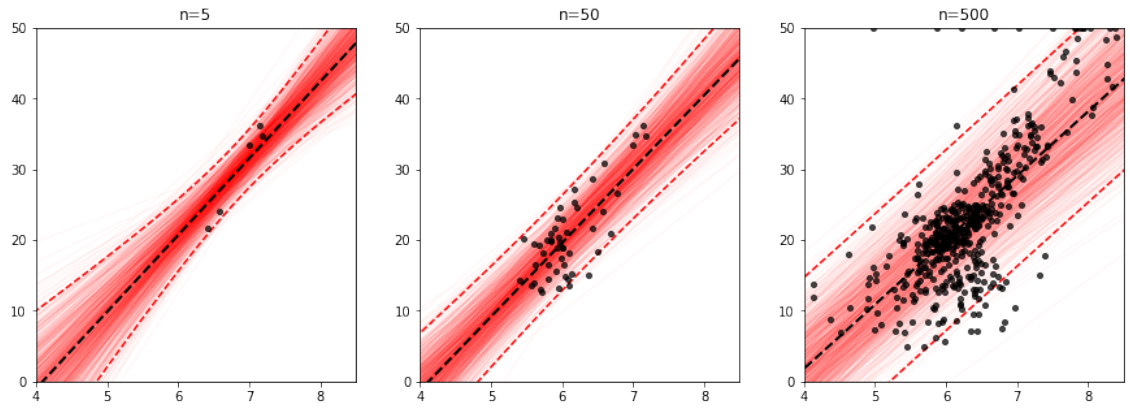

Figure 6: **Example of *NIW* predictive posterior distribution.**

2.6 Bayesian Non-linear models

Sometimes phenomena in nature are more complex than just the result of linear interactions. A more flexible alternative to linear methods are models that present nonlinearities (in parameters) capable of discovering the complex relationships between multiple variables. This group includes many of the most advanced families of algorithms, such as support vector regression, Gaussian processes or artificial neural networks.

A nonlinear model applies nonlinear transformations to the input space, so a new representation (feature space) where the observations are linearly arranged can be obtained. This type of models generalises linear models, as they are capable of expressing a more extensive range of functions. However, the high degree of freedom that this flexibility provides, also makes the fitting process a more computationally heavy problem.

In the Bayesian context, nonlinear regression is represented by a wide range of models, but most of them rely on Monte Carlo inference methods to approximate

the posterior and predictive distributions. For instance, a Bayesian approach to neural networks (BNN) uses Monte Carlo dropout sampling [GG16] to obtain a distribution of outputs from which we can estimate the uncertainty of the model.

In this project, we focused our attention on Gaussian processes, an alternative to other sampling methods (in the assumption of a Gaussian likelihood) from which we can obtain high performance in the regression task, while also at the same time can provide accurate measurements of the model uncertainty.

2.6.1 Basis functions

A simple technique that can be used to start introducing nonlinearities to our model is *basis function expansion*. A *basis function* ϕ is defined as a set of transformations that can be applied to our input vector to obtain a new representation x_ϕ that introduces a certain type of nonlinearities, defined by the type of function.

A typical example of a basis expansion is the obtained using a *polynomial basis function*, which transform the input vector x to a set of covariates of different polynomial order:

$$\phi(X) \in \mathbb{R}^{p_\phi} \ ; \ \phi(X) = [1, x, x^2, x^3, ..., x^{p_\phi}]. \tag{23}$$

With this new transformed input space, a *linear regression* model will be able to also capture some of the non-linear relationships of the original variables:

$$y = \phi(X)^\top w + \epsilon \tag{24}$$

2.6.2 Gaussian processes

A Gaussian process (GP) [RW05] is a particular type of nonparametric model that, following the Bayesian methodology, can provide a posterior distribution from which we can sample all the possible functions that are fitted to the observed data. These functions can be used as regression lines when fitting a curve and can be extended, by using a logistic function, to perform binary probabilistic classification.

The idea of GPs is based on the infinite-dimensional generalisation of the multivariate normal distribution. Equivalently to how the MVN distribution is parametrized by its mean vector μ_k and covariance matrix Σ_d, the GP is defined by its mean

function $m(x)$ and covariance function $k(x, x')$, as follows:

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))], \end{aligned} \tag{25}$$

so the Gaussian process is expressed as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{26}$$

We can see how this definition of GPs states a distribution over functions f, which can be used for Bayesian regression. Each of the generated functions defines a set of random variables $f(X) = (f(x_1), ..., f(x_n))^T$, so the evaluation of f at each location x_i maps to the value y_i.

Then, we can define the Gaussian process regression (GPR) model as follows:

$$\begin{aligned} y_i &= f(x_i) + \epsilon_i \\ f &\sim GP(0, K) \\ \epsilon_i &\sim \mathcal{N}(0, \sigma^2) \end{aligned} \tag{27}$$

where predictions are expected to be affected by some Gaussian noise ϵ, and the prior $p(f)$ has been assumed to has a zero mean function $m(x)$, and a covariance matrix K, obtained by evaluating the covariance function $k(x, x')$ for each pairs of observations. From the different types of covariance functions that can be specified, we will focus on using the *Radial Basis Function* (RBS) kernel, that as stated in [RW05] corresponds to a Bayesian linear regression model with an infinite number of basis functions. The RBF function is defined as follows:

$$k(x, x') = \sigma_f^2 exp\Big(\frac{-(x - x')^2}{2l^2}\Big), \tag{28}$$

with hyperparameters σ_f^2 and l, which define the scale and the bandwidth of the covariance, respectively. During the training process, the hyperparameters of the kernel are also optimised.

Once we have defined the prior $p(f)$, we can use Bayes' rule to obtain the posterior distribution. By conditioning on data we will obtain a new parametrization of the GP, where the mean and the covariance function is updated as follows:

$$p(f|y) \sim \mathcal{N}(K(K + \sigma^2 I)^{-1} y, K - K(K + \sigma^2 I)^{-1} K) \tag{29}$$

Similarly, we can obtain the *posterior predictive distribution* for a new set of datapoints X^*, with:

$$p(f^*|y) \sim \mathcal{N}(K_{X_{n+1}X}(K_{XX} + \sigma^2 I)^{-1} y, K_{X_{n+1}X_{n+1}} - K_{X_{n+1}X}(K_{XX} + \sigma^2 I)^{-1} K_{XX_{n+1}}) \tag{30}$$

As we can see in Fig. 7, once we update the prior, we can use the posterior distribution to sample functions f^* that are consistent with the observed datapoints (red solid lines). The model is able to capture the noise of the observations by adding a white noise kernel to the covariance function and optimizing its hyperparameter value during training, as it is seen in the center a right panel.

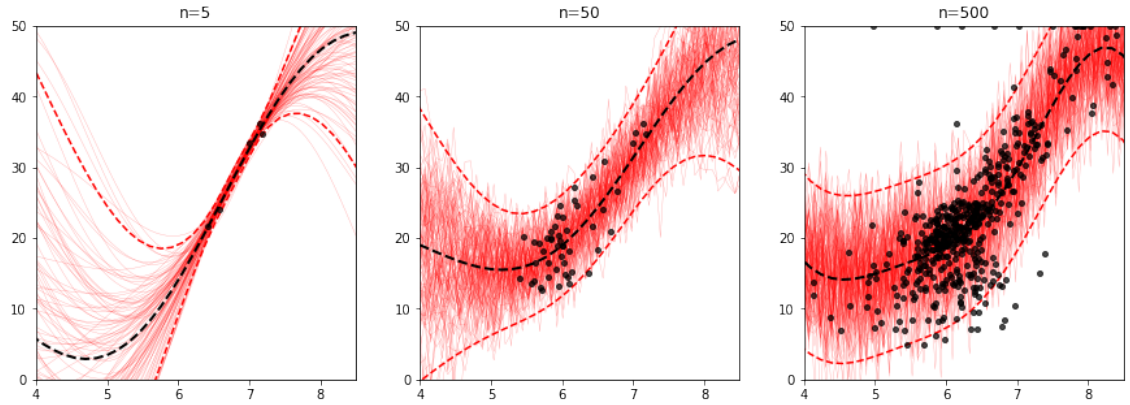

Figure 7: **Example of Gaussian process predictive posterior distribution.**

However, the main problem associated with GP is their lack of scalability. The associated complexity time of a GP during the fitting process is $O(n^3)$, and its complexity space is $O(n^2)$. Due to its performance demands, GPs present unfeasible training and testing time regarding the performance requirements of this project. As an alternative, we can explore methods that approximate the performance of GP but with more tractable complexity bounds.

2.6.3 Sparse Gaussian processes

An approximation method to GP is what is known as *sparse Gaussian processes* (SGP) [QCR05]. This regression model can approximate the posterior distribution over functions by defining a small set of pseudo-points from the input space, that

can be used to estimate a compressed fitting similar to the obtained with exact GP inferences. As stated in [QCR05], the inference process performed in SGP can be interpreted as exact inference with an approximated prior.

Formally, the SGP model selects a small subset of size m from the input space, with $m << n$, reducing the training complexity time to $O(nm^2)$, and the testing time $O(m^2)$ per test case. The use of a limited number of pseudo-point reduces the memory and computational demand of the algorithm significantly, while still providing a similar accuracy to the full GP model.

In Fig.8, we can see noise-free samples of the latent function of the SGP model (used library does not allow for sampling directly from the predictive posterior distribution). We also visualise the location in the input space of the inducing points (blue dashed lines). The location of these inducing points is optimised during the training process, and as can be seen, the obtained 95% confidence intervals are very consistent with the margins obtained with GP in Fig. 7.

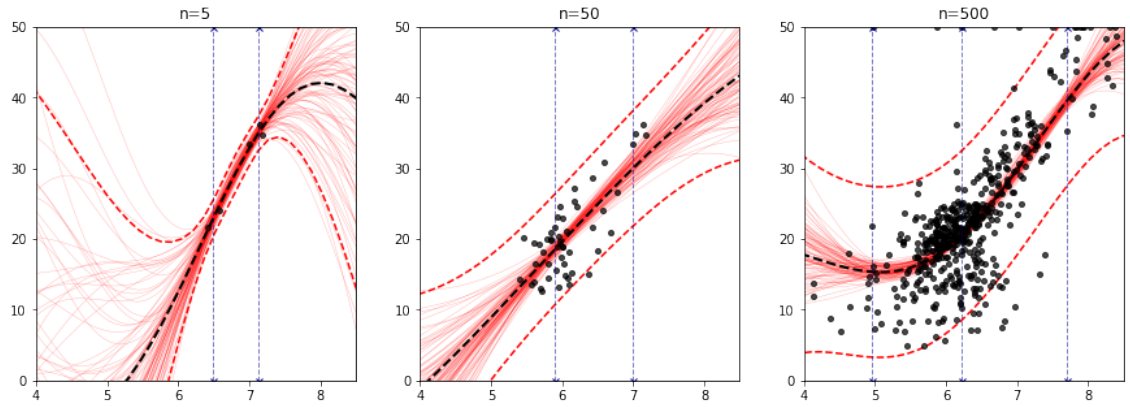

Figure 8: **Example of sparse Gaussian process model.**

The Bayesian fitting process of the SGP is similar to the presented in the previous section. However, differences exist on the type of method used in the optimisation of the inducing points and the kernel hyperparameters. Most used SGP implementations are based on two main optimization techniques: Fully Independent Training Conditional (FITC) [SG06]; and Variational Free Energy (VFE) approximation [Tit09]. Differences between both approximation methods lead to different theoretical and practical properties, as discussed in [BvdWR16].

Over the years, progress in the literature has introduced better SGP implementa-

tions with improved complexity bounds. For instance, [CBFH15] presents a new efficient sparsification algorithm, based on VFE, which optimises a single objective for selecting the inducing points jointly and the GP hyperparameters, outperforming previous methods on discrete datasets. Alternatively, [CB17] proposes an implementation called decoupled Gaussian processes (DGPs), that decouples the set of m points into two sets m_α and m_β, where $m_\beta << m_\alpha$, that are independently used to model the mean and the covariance functions. Decoupling the set of inducing points reduces the complexity bounds to $O(npm_\alpha + nm_\beta^2 + m_\beta^3)$ in time and $O(nm_\alpha + m_\beta^2)$ in space. Recently, [PGWW18] has been presented, combining also advancements from [WN15], proposing a system called KISS-GP (w/ LOVE), that can train a SGP with complexity time $O(k(n + m \log m))$ and space $O(km)$, with $k << m$ referring to the number of iterations required for optimization. Moreover, this implementation allows computing the predictive posterior variance in constant time $O(k)$ per test case. Find in Table 1 a summary comparing the complexity bound of these techniques concerning its training complexity time and space, and complexity time of computing its predictive variance per test case.

To compare the performance of the different models, we selected the method of [Tit09], which is implemented by the Python library *GPFlow* [MvN$^+$17], although as we discuss in Sec. 4, it would be needed to explore different implementations of SGP which better meets the system requirements.

Method	**Train time**	**Train space**	**Test time**
GP [RW05]	$O(n^3)$	$O(n^2)$	$O(n^2)$
Sparse GP [Tit09]	$O(nm^2)$	$O(m^2)$	$O(m^2)$
Sparse GP [CBFH15]	$O(nm^2)$	$O(m^2)$	$O(m^2)$
Decoupled GP [CB17]	$O(npm_\alpha + nm_\beta^2 + m_\beta^3)$	$O(nm_\alpha + m_\beta^2)$	–
KISS+LOVE [PGWW18]	$O(k(n + m \log m))$	$O(km)$	$O(k)$

Table 1: Comparison of different GP implementations.

2.7 Evaluation methods

In order to compare the proposed models, we need to specify correct evaluation methods. For the goal of this project, a valid model should be evaluated regarding accuracy and how well calibrated their confidence intervals are.

2.7.1 Accuracy

To evaluate the accuracy of the model we need to specify a method that measures how close are our predictions to the ground truth. The selected metrics are *Mean Squared Error (MSE)* and the *Pearson correlation coefficient* between the predicted vector Y_{pred} and the ground truth vector of barrier values Y.

Pearson correlation coefficient measures what is the linear dependence between two variables. Its value ranges from -1 (variables are indirectly correlated) to 1 (variables are directly correlated), meaning a 0 value that variables are completely uncorrelated. This is computed as the fraction of the empirical covariance and the standard deviation of the variables:

$$R_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{31}$$

The correlation coefficient is calculated between the predicted values Y_{pred} and the actual values Y so the closer the correlation is to 1, the better are the predicting capabilities of the model. A visual evaluation based on the same idea can be performed by drawing a scatter plot of real values vs predicted values.

2.7.2 Uncertainty calibration

To evaluate how well calibrated our model is, regarding the uncertainty of the predictions, we need to compute a metric that assess the quality of the predictive posterior confidence intervals.

Once we train a certain model, we can obtain the predictive variance and use it to design predictive intervals. For instance, a model whose predictive distribution is known to be distributed as a Gaussian should find 95% of the new observations inside the $\mu_x \pm 1.96\sigma_x$ interval.

Then, we can estimate what the proportion of test data that lie inside those intervals is. We will say that a model is well calibrated when the computed proportion

approaches the theoretical confidence level of the interval:

$$C_{95\%} = \frac{n_{95\%}}{n} * 100, \tag{32}$$

where $n_{95\%}$ are the number of test points inside the 95% confidence level interval.

3 Results

In this section, we will present the results obtained by comparing all the proposed models in the task of predicting energy barriers. Then, we will show more insights into how the selected model performs on atomic simulations and compare it to the current neural network.

3.1 Data

The dataset provided by the Helsinki Institute of Physics [LJV+18b] includes 11 579 102 samples of Copper atoms configurations, represented with the encoding described above, and for which the barriers have been accurately calculated by the NEB method. Additional to this dataset, a collapsed version is provided, where symmetrical rotations of each configuration have been removed. In the three-dimensional space, a configuration can have four equivalent rotations, and therefore, the size of the original dataset can be potentially reduced by one-quarter of its size. This is true for the provided collapsed dataset, which includes 2 915 901 observations. Both collapsed and non-collapsed dataset group the input configurations by their surface structure type (i.e. 100, 110 and 111).

The data has been obtained through the nudged elastic band (NEB) method [JMJ98]. For a given initial and final state of an atom in the atomic structure, this method allows calculating the intermediate steps in the minimum energy transition path, and its associated energy barrier. Initially, this method assumes an artificial path (e.g. linear transition of the atom) that will be optimised to approximate intermediate configurations to the minimum energy path. NEB applies spring forces between adjacent configurations, so each step is constrained to remain between the previous and following configurations. The saddle point of the energy gradient in the minimum energy path defines the energy barrier value for that transition. It is

possible to converge to this point with an NEB variant called *climbing image* [HJ00], which takes the highest energy point and move it along the transition path to the saddle-point configuration, where energy is maximised.

3.2 Prediction accuracy

We compare all the models presented in Sec. 2.5 and Sec. 2.6 in terms of the accuracy and uncertainty calibration.

In particular, we show results for the normal-Gaussian (NGa) model with fixed variance $\sigma^2 = 0.1$, the normal-inverse gamma (NIG) and normal-inverse Wishart (NIW) models, all of them evaluated with and without polynomial basis expansion of degree 2 with interactions. Non-linear models are represented by the standard Gaussian process (GP), based on the implementation included in the Python library *sklearn* [PVG+11], and the sparse Gaussian process (SGP) model, implemented by the library *GPFlow* [MvN+17]. The kernel function selected for the GP is the *Radial Basis Function* (RBS) kernel with additive white noise. We evaluate SGP for a different number of inducing points $m = 10, 50, 100$.

Train and test sets are generated from a random 80/20 split. Due to the scalability limitations of the standard GP, we train the models in a small subset of the original dataset of 10 000 observations and evaluate them on a test set of 2500 observations.

We train and evaluate all the models for the three different surface types $\{100\}$, $\{110\}$, $\{111\}$ and also for a set of the same size that includes all the surfaces randomly mixed $\{All\}$. Also, we perform tests for both collapsed and non-collapsed versions of the datasets. You can find in Table 2 and Table 3 a summary of the obtained results:

We can see from the results how GPs outperforms all the other models, generating predictions with the lower error rates, which are also well calibrated for most of the scenarios. However, as we previously discussed, GPs are not suitable due to its lack of scalability. As an alternative, SGP appears as a valid substitute of these models, as we can see how it approximates the performance of the GPs when the number of inducing points m is increased. For a set size $m = 100$ we see how the model ranks as the best alternative to GPs in all the evaluations, incurring in a much lower computational cost.

	{100}			{110}			{111}			{All}		
	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$
NGa	0.04089	0.818	99.48	0.03410	0.824	99.4	0.03308	0.821	99.92	0.03750	0.814	99.48
NGa + Poly(2)	0.01660	0.930	100	0.01937	0.904	100	0.02225	0.883	100	0.02216	0.894	99.92
NIG	0.04089	0.818	95.92	0.03409	0.825	93.76	0.03308	0.821	95.76	0.03750	0.814	95.28
NIG + Poly(2)	0.01665	0.930	96.44	0.01950	0.904	99.68	0.02225	0.883	93.4	0.02217	0.894	**95.04**
NIW	0.04089	0.818	**95.84**	0.03409	0.825	93.56	0.03308	0.821	95.36	0.03750	0.814	95.08
NIW + Poly(2)	0.01665	0.930	96.2	0.01950	0.904	99.64	0.02225	0.883	92.68	0.02217	0.894	94.76
GP	**0.00755**	**0.969**	96.12	**0.01309**	**0.936**	92.76	**0.01958**	**0.898**	91.68	**0.01452**	**0.932**	95.08
SGP (m=10)	0.01785	0.925	98.88	0.02143	0.893	95.6	0.02537	0.866	96.36	0.02458	0.882	97.04
SGP (m=50)	0.01627	0.932	98.2	0.0181	0.911	94.4	0.02276	0.881	**94.76**	0.02262	0.892	95.92
SGP (m=100)	0.01365	0.943	98.44	0.01631	0.920	**95.04**	0.02258	0.882	94.72	0.02220	0.894	95.88
Average	0.02278	0.902	97.55	0.02295	0.885	96.384	0.02563	0.864	95.46	0.02630	0.873	96.348

Table 2: Comparison of models trained on non-collapsed data.

	{100}			{110}			{111}			All		
	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$
NGa	0.04064	0.809	99.52	0.03067	0.847	99.76	0.03416	0.806	99.96	0.03706	0.812	99.28
NGa + Poly(2)	0.01793	0.920	100	0.0132	0.937	100	0.02406	0.868	100	0.02059	0.900	99.8
NIG	0.04063	0.809	95.84	0.03063	0.847	93.76	0.03416	0.806	93.76	0.03706	0.812	**94.8**
NIG + Poly(2)	0.01799	0.920	93.68	0.01346	0.936	95.4	0.02410	0.868	99.72	0.02060	0.900	94.28
NIW	0.04063	0.809	95.6	0.03063	0.847	93.32	0.03416	0.806	93.36	0.03706	0.812	94.68
NIW + Poly(2)	0.01799	0.920	92.92	0.01346	0.936	**94.68**	0.02410	0.868	99.68	0.02060	0.900	93.88
GP	**0.00861**	**0.962**	**94.6**	**0.00720**	**0.966**	92.72	**0.02011**	**0.891**	91.04	**0.01165**	**0.945**	94.24
SGP (m=10)	0.01814	0.919	97.92	0.01638	0.921	96.4	0.0256	0.858	**94.64**	0.0225	0.890	96.56
SGP (m=50)	0.01384	0.939	97.84	0.01228	0.941	95.88	0.02382	0.869	92.64	0.02092	0.899	95.68
SGP (m=100)	0.01270	0.944	97.8	0.01125	0.946	95.96	0.02368	0.870	92.12	0.01930	0.907	95.96
Average	0.02291	0.896	96.57	0.0179	0.913	95.788	0.02680	0.8513	95.692	0.02473	0.878	95.916

Table 3: Comparison of models trained on collapsed data.

Something noticeable is how linear models with polynomial expansion are also able to provide a good performance regarding accuracy and calibration (here NGa model is not included). Probably this is because the underlying function we are trying to approximate is simple enough that to be handled by linear models with simple non-linearities. However, we could expect that the expressiveness of these models

might be limited in the future when the complexity of the task increases once we start using configurations with different atoms type.

We can also analyse from the tables, what is the differences in performance concerning whether we are using the collapsed or non-collapsed version of the dataset during training. If we compare how the models perform on average by dataset used (last row on tables), we can see that by using the collapsed representation, we can obtain a slight improvement in accuracy and calibration.

If we compare the results per surface type, it can be noticed how training on the surface {111} reports an evaluation with higher errors. Also, it can be seen that we obtain better performance if we combine the models trained by their surface type independently, than by feeding one model with all the data combined. This is consistent to the results reported in [LJV+18a], where they use an *ANN* classifier with very high accuracy in predicting what is the surface type of the given configuration and using this to select the corresponding model and improving the accuracy of the overall predictions.

3.3 Kinetic Monte Carlo Simulations

Once we have identified the sparse Gaussian process as a suitable model, we can perform some tests and compare its performance with the current developed neural network.

We trained the SGP model with a set of inducing points of size $m = 100$, for the three surface types. Train and test set are generated from a random split 80/20 of the whole collapsed dataset: 287 870 samples from surface {100}, 378 364 samples from surface {110} dataset and 2 249 667 samples from surface {111}.

The results obtained from evaluating these models are summarised in the following table:

	{100}			**{110}**			**{111}**		
	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$	MSE	R	$C_{95\%}$
SGP (m=100)	0.00417	0.9845	97.73	0.00328	0.9830	97.81	0.0089	0.9558	95.75

Table 4: SGP model trained in collapsed dataset.

We can see in Table 4 how by training our model in a larger dataset can provide a remarkable improvement in the model performance. If we compare this table with the results of the model SGP($m = 100$) in Table 2 and Table 3, we can see an increment in the accuracy of the predictions: MSE values get reduced to a third, and R coefficients increase from an average value of 0.915 to 0.974.

Regarding calibration, the model does not show up such an improvement, but still, it can provide predictive intervals with an average confidence level of 97%, very close to the theoretical 95% confidence level.

We can see in Fig. 9, how the trained model can produce distributions of barriers very similar to the ground truth barriers set. The first row of the figure illustrates the distributions of the predictions as the model gives them. We can see how the distributions have a left tail that corresponds to negative barriers values.

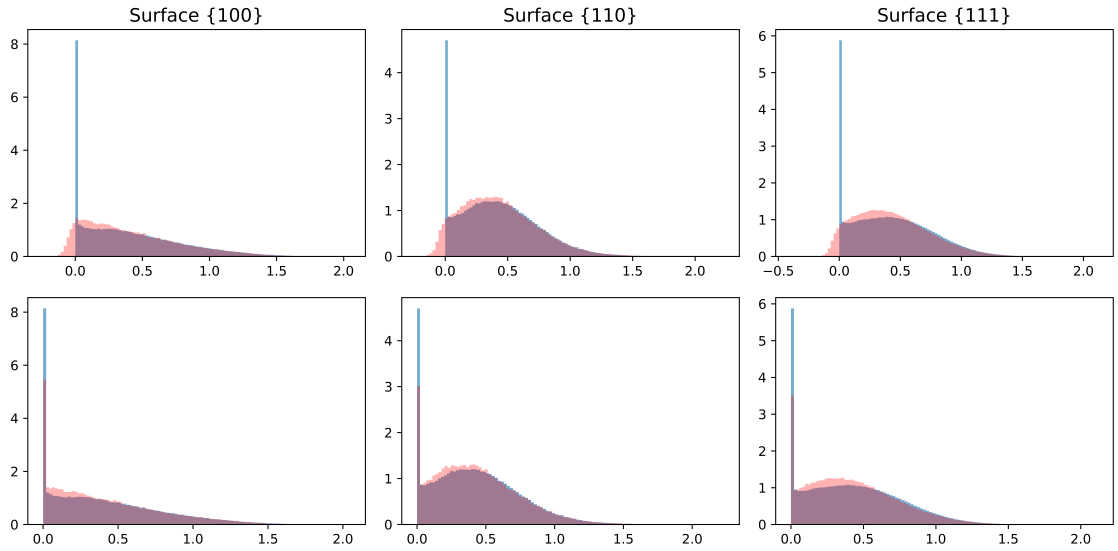

Figure 9: **Comparison between predicted barriers (red) and real barriers (blue) distributions.**

To make these predictions consistent with the natural laws of physics, we need to truncate the results of predictions to a minimum value of 0. Then, we can see in the second row of Fig. 9, how after truncating the predictions, we can finally obtain an almost identical distribution in which the large peaks of zero values are also presented. This post-processing of the results helps to increase in a small amount the accuracy of the model.

Then, we can proceed to compare the performance of the SGP (m=100) model with the existing neural network. We use in combination with the SGP model, an artificial neural network developed in [LJV+18a] for surface type classification (we reference this network as ANN_c to make a distinction with the ANN used for barriers predictions). We use this network to classify input atoms configurations by its surface structure type, and then to select the corresponding trained SGP model.

In Fig. 10 we illustrate the correlation plots between the computed and predicted barriers for both, the current implemented ANN and the SGP model. Each predicted barrier is computed by using the trained models shown in Table 4 in combination with the ANN_c classifier. The resulting plots show that the correlation distribution of the SGP is practically equivalent to the results obtained from the ANN model, with an almost identical RMSE value: for ANN is 0.086 and for $SGP + ANN_c$ is 0.088.

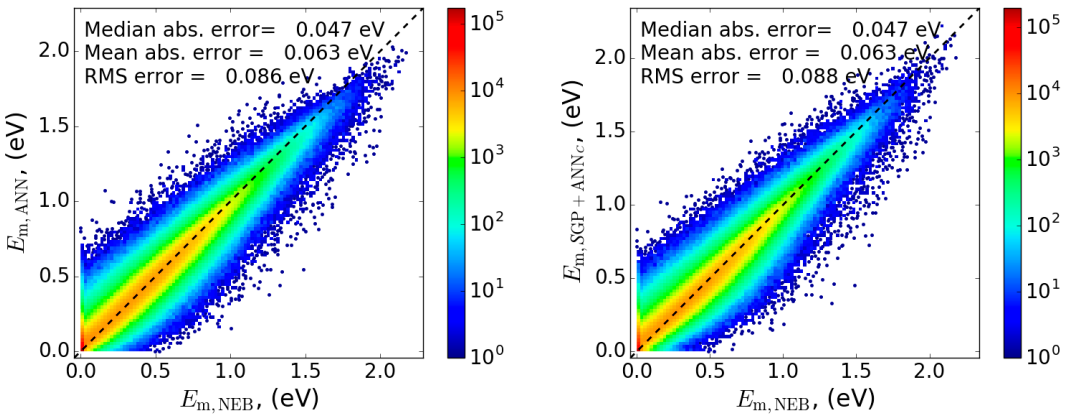

Figure 10: **Correlation between the computed and the predicted barrier values. Left: Predictions from the *ANN*. Right: Predictions from the SGP.**

Finally, we asked the Helsinki Institute of Finland to use the trained SGP models to feed the kinetic Monte Carlo method, and to assess if the predicted energy barriers produce consistent atomic simulations compared to empirical experiments. Figure 11 shows the initial and final state of running KMC simulations for both models.

Their evaluation suggests that simulations run as expected for all the surfaces types, even producing a more natural behaviour on the final state of surface 110 simulation

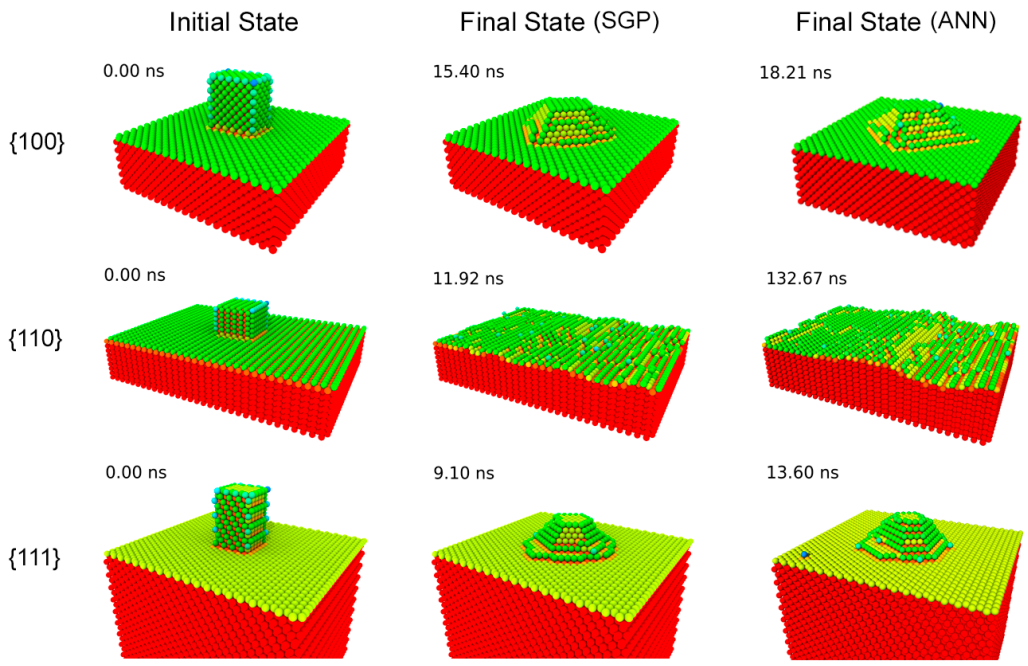

Figure 11: **Initial and final state of KMC simulations.**

if we compare to the simulations obtained from the *ANN* model. This can be noticed in Fig. 11, where some artefacts that are not expected can be identified in the surface of the *ANN* simulations, while SGP simulations show a more uniform flatten surface.

4 Conclusions

In this thesis we have presented a new approach, based on sparse Gaussian process, to predict the potential energy barriers of migrating atoms. This method not only allows us to obtain predictions with a high accuracy equivalent to the one obtained with the currently used models, based on artificial neural networks, but the model is also able to asses its uncertainty respect to the predictions performed.

The application of SGP contributes to developing the proposed active-learning-based atomic simulator, which can use the uncertainty assessment capabilities of the model to request the necessary labelled data at minimum cost. We have also shown how the performance of the selected algorithm is bounded by a tractable complexity

time, so barriers predictions can be performed without producing a bottleneck in the system.

Future work on this task should evaluate how the proposed model generalises to more complex instances of the problem when configurations include more than one atom type. Also, alternative implementations of SGP that fulfil the requirement of online training should be evaluated, a key component of the proposed simulator system. Examples of these implementations are found in works like [BvWSV15] or [BNT17].

We expect that the Machine Learning methods proposed in this work help to reduce the incurred computational complexity of new materials research.

References

AFHW91 A. Fichthorn, K. and H. Weinberg, W., Theoretical foundations of dynamic monte carlo simulations. In *J. Chem. Phys.*, volume 95(2), July 1991, pages 1090–1096.

BNT17 Bui, T. D., Nguyen, C. and Turner, R. E., Streaming sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, volume 30, Curran Associates, Inc., 2017, pages 3299–3307.

BvdWR16 Bauer, M., van der Wilk, M. and Rasmussen, C. E., Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, volume 29, Curran Associates, Inc., 2016, pages 1533–1541.

BvWSV15 Bijl, H., van Wingerden, J.-W., Schon, T. B. and Verhaegen, M., Online sparse Gaussian process regression using FITC and PITC approximations. In *IFAC-PapersOnLine*, volume 48, 2015, pages 703–708.

CB17 Cheng, C.-A. and Boots, B., Variational inference for gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, volume 30, Curran Associates, Inc., 2017, pages 5184–5194.

CBFH15 Cao, Y., Brubaker, M. A., Fleet, D. J. and Hertzmann, A., Efficient optimization for sparse gaussian process regression. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 37, December 2015, pages 2415–2427.

GG16 Gal, Y. and Ghahramani, Z., Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, volume 48 of *ICML'16*, JMLR.org, 2016, pages 1050–1059.

Her16 Hermann, K., Crystallography and surface structure: An introduction for surface scientists and nanoscientists. Wiley, 2016.

HJ00 Henkelman, G. and Jónsson, H., Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. In *The Journal of Chemical Physics*, volume 113, 2000, pages 9978–9985.

JMJ98 Jónsson, H., Mills, G. and Jacobsen, K. W., Nudged elastic band method for finding minimum energy paths of transitions. In *Classical and Quantum Dynamics in Condensed Phase Simulations*, June 1998, pages 385–404.

LJV+18a Lahtinen, J., Jansson, V., Vigonski, S., Baibuz, E., Domingos, R., Zadin, V. and Djurabekova, F., Artificial neural networks for Cu surface diffusion studies. In *ArXiv e-prints*, June 2018.

LJV+18b Lahtinen, J., Jansson, V., Vigonski, S., Baibuz, E., Domingos, R., Zadin, V. and Djurabekova, F., Data sets and trained neural networks for Cu migration barriers. In *ArXiv e-prints*, June 2018.

MvN+17 Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., Leon-Villagra, P., Ghahramani, Z. and Hensman, J., GPflow: A Gaussian process library using TensorFlow. In *Journal of Machine Learning Research*, volume 18, April 2017, pages 1–6.

PGWW18 Pleiss, G., Gardner, J. R., Weinberger, K. Q. and Wilson, A. G., Constant-Time Predictive Distributions for Gaussian Processes. In *ArXiv e-prints*, March 2018.

PVG+11 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., Scikit-learn: Machine learning in Python. volume 12, 2011, pages 2825–2830.

QCR05 Quinonero Candela, J. and Rasmussen, C., A unifying view of sparse approximate gaussian process regression. In *Journal of Machine Learning Research*, volume 6, December 2005, pages 1935–1959.

RAM06 Rossi, P., Allenby, G. and McCulloch, R., Bayesian statistics and marketing. Wiley, October 2006.

RW05 Rasmussen, C. E. and Williams, C. K. I., Gaussian processes for machine learning. MIT Press, 2005.

Set09 Settles, B., Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

SG06 Snelson, E. and Ghahramani, Z., Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18, MIT Press, 2006, pages 1257–1264.

ST17 Shwartz-Ziv, R. and Tishby, N., Opening the Black Box of Deep Neural Networks via Information. In *ArXiv e-prints*, March 2017.

Tit09 Titsias, M. K., Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, volume 12, 2009, pages 567–574.

WA10 Walter, G. and Augustin, T., Bayesian linear regression – different conjugate models and their (in)sensitivity to prior-data conflict. In *Statistical Modelling and Regression Structures: Festschrift in Honour of Ludwig Fahrmeir*, Physica-Verlag HD, 2010, pages 59–78.

WN15 Wilson, A. G. and Nickisch, H., Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *ArXiv e-prints*, March 2015.